MJRET

Open Access

Multidisciplinary Journal of Research in Engineering and Technology

Journal homepage: www.mjret.in

ISSN:2348 - 6953

OPTIMIZATION OF NETWORK BANDWIDTH OVER INTERNET THROUGH LOW BANDWIDTH NETWORK FILE SYSTEM (LBFS)

Bodhe Yogesh Uttamrao
Assistant Professor, Department of Computer Engineering
Dr. D. Y. Patil School of Engineering
Pune, India

Abstract: *This Paper presents generalized approach to low bandwidth network file system i.e. LBFS which is designed for low-bandwidth networks. Low bandwidth leads to slow network performance. LBFS focuses on similarities between different files or different versions of the same file to save bandwidth. Duplicate data transfer is avoided which leads to less consumption of bandwidth over the network. Merits & demerits of existing system are discussed and new system design is proposed which can be implemented in different types of networks.*

Keywords*: Network file system, LBFS, low-bandwidth networks, network performance*

1. INTRODUCTION

A distributed file system is a collection of loosely coupled computers interconnected by a communication link. From the point of view of specific computer in a distributed system, hosts & their respective resources are remote, whereas its own resources are local. Important part of distributed operating system, a file system provides file services to clients. A client interface for a file is formed by set of primitives, called file operations, such as open a file, remove a file, read from a file, write to a file & so on. A distributed system is a file system whose clients, servers & storage devices are dispersed among the interconnected computers of a distributed system. In practice, the concrete configuration & implementation of a distributed file system may vary and it is difficult to determine the best implementation. A distributed file system can be implemented as a part of distributed operating system or by a software layer whose primary function is to manage the communication between conventional operating systems & file systems.

Some examples of distributed file systems are Sun's *Network File System* (NFS) built on remote procedure call mechanism, *The Andrew File System* (AFS) based on whole-file

caching, *Bayou File System* to manage update conflicts, *Coda File System* for disconnected operations, *Concurrent Version System* (CVS) for version management*, Echo System* designed around truly global namespace, *Ocean Store* - An Extremely Wide Area Storage System, *Rsync* - File Synchronization System, *Self-certifying File System* (SFS), File System for *JET* Data, *Leases* on Clustered File system & so on.

2. RELATED WORK

LBFS [1] is a client server based network file system that is optimized for low bandwidth networks. Main novelty is identifying duplicates in data through content-defined chunking and not transmitting these duplicates between the client and the server. The LBFS uses a lot of other existing bandwidth optimization techniques and could potentially benefit from lot of techniques for bandwidth reduction especially over wide-area networks. The important aspects of LBFS are:

2.1 Content defined chunking to identify duplicates:

The technique of splitting a file into smaller units (chunks) for purpose of comparing for identifying duplicates is called chunking. A simple chunking algorithm is to chunk a given file into chunks of equal sizes called fixed sized blocking.

2.2 Indexing:

On both the client and server, LBFS must index a set of files to recognize data chunks so that it can avoid sending of duplicate data chunks over the network.

2.3 Compare by Hash:

This is hashing technique used in LBFS to identify the duplicate chunks. Before sending a chunk, the sender first transmits the hash of the chunk to the receiver. The receiver checks to see if it has a local chunk with the same hash value. If it does, it assumes that it is the same chunk as the sender's, without actually sending chunk at receiver side.

2.4 Chunk Database:

LBFS uses a database to identify and locate duplicate data chunks. It indexes each chunk by hash value.

2.5 LBFS Protocol:

The LBFS protocol is based on NFS version 3.

3. SYSTEM IMPLEMENTATION

3.1 Server Architecture & Client Architecture:

The login is provided to authorize the user. It is a way to provide a security to the system & avoid the unauthorized access to data by any unauthorized user. Chunking is used to divide the file into chunks. Chunking is base of the system. The file is sent through chunks. The naming system is developed to name the chunks uniquely.

3.2 Server Chunk Database & Client Chunk Database:

Creation of chunk database is the feature of our system. The chunk database is created & maintained using chunking. Using chunk database & hash function the index database is created. The unique location is provided to chunk database. It is necessary for storage purpose & useful while sending & receiving chunks i.e. in client-server communication.

3.3 Creation Of Hash Values And Index Database:

Hash function which used in the system is SHA-512 [4]. It is the one of the most secure cryptographic hash function. It is used to create hash values. Hash value is nothing but index value for the chunk. For each chunk separate index is created. Using chunk database & hash function, index database is created at both client & server side. The index database is used in client-server communication. While transferring the file, first the indexes will be sent. It is also used for the comparison purpose. The indexes will be compared first & depends on that the chunks will be sent or received.

3.4 Rebuilding File From Chunks:

After transferring the chunks, it is necessary to rebuild the chunks into one single file. The rebuild option will rebuild the chunks into one single file. Rebuilding the chunks will take chunk file name as input & rebuild all chunks into single file.

3.5 System Architecture:

The pictorial representation of system architecture shows client-server communication.

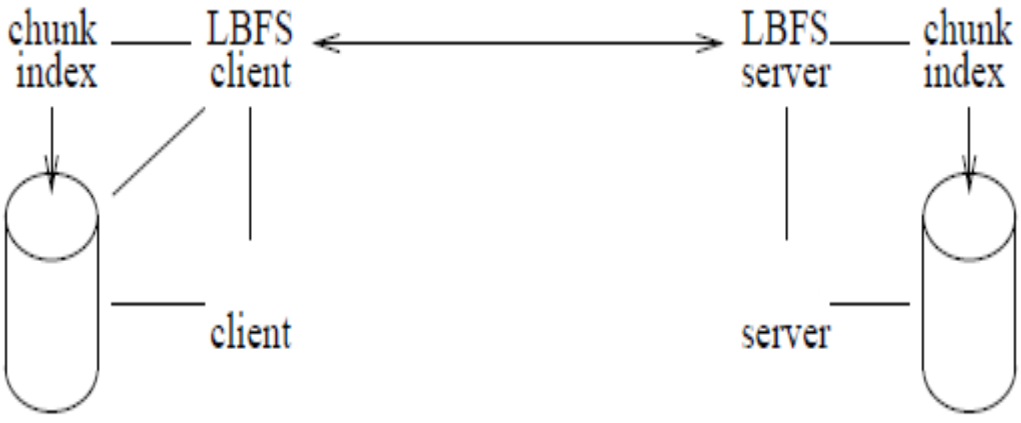

Figure1. System Architecture

4. OBSERVATIONS & RESULTS:

4.1 Read Operation (Downloading):

Here we are taking three types of file into consideration. These are of type Video file, Audio file & Text file. In read operation data is downloaded from server to client when client wants to read data from server. Server is sending file to client at various conditions & bandwidth is calculated for every file transfer. The bandwidths are compared between traditional file system and low bandwidth network file system & bandwidth saving is calculated.

Three types of files are considered

- Video file: 20 MB
- Audio file: 5 MB
- Text file: 2 MB

Following operations are done & bandwidth is calculated.

- Norm: file is directly sent from server to client
- Cmprs: file is compressed first & sent from server to client
- Lbfs1- file is divided into chunks & sent using lbfs, no similar data
- Lbfs2- file is compressed, divided into chunks & then sent using lbfs, no similar data
- Lbfs3- file is divided into chunks & sent using lbfs, similar data
- Lbfs4- file is compressed, divided into chunks & then sent using lbfs, similar data

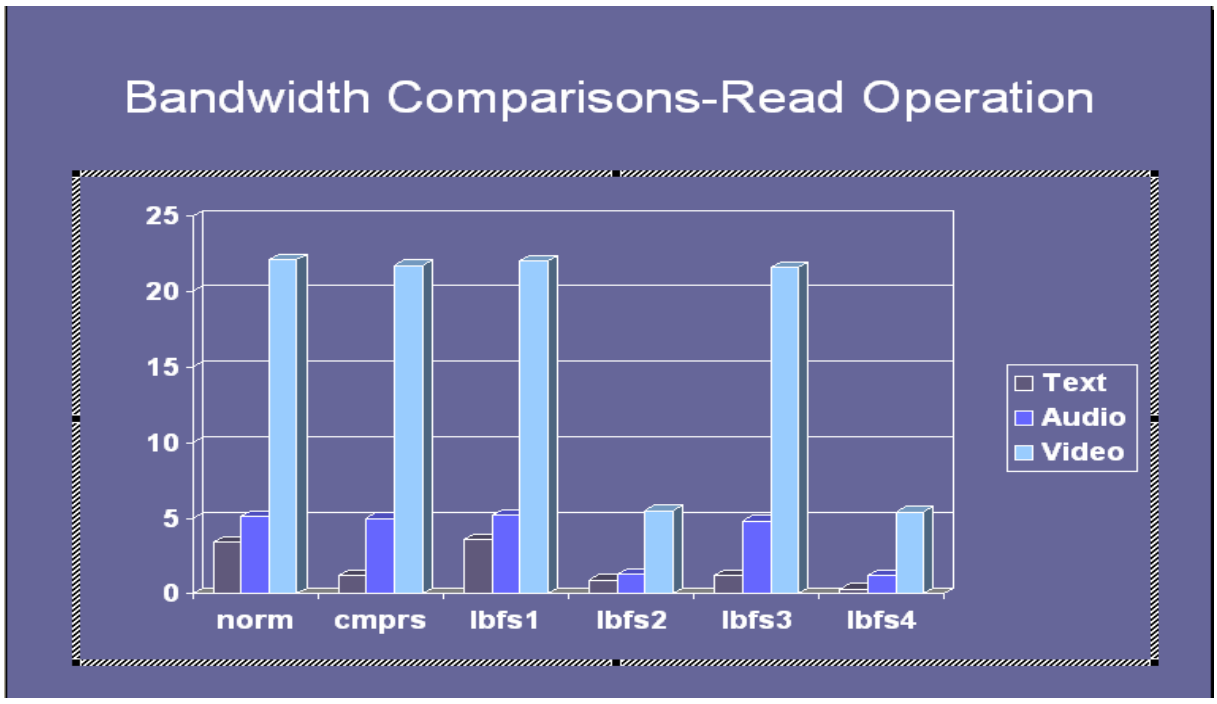

Figure. 2 Bandwidth Comparisons-Read Operations

4.2 Write Operation (Uploading):

Here we are taking text files into consideration. In write operation data is uploaded from client to server when client wants to write data to server. Server is receiving file from client at various conditions & bandwidth is calculated for every file transfer. The bandwidths are compared between traditional file system and low bandwidth network file system & bandwidth saving is calculated.
Text files are considered

- Text file: 2 MB

Three cases are considered:

1. Best case: file is written at the end
2. Average case: file is written in between
3. Worst case: file is written at the start

Following operations are done & bandwidth is calculated:

- Norm: file is updated & directly sent from client to server
- Cmprs: file is updated first, compressed & sent from client to server
- Lbfs1- file is updated, divided into chunks & sent using lbfs
- Lbfs2- file is updated, compressed, divided into chunks & then sent using lbfs

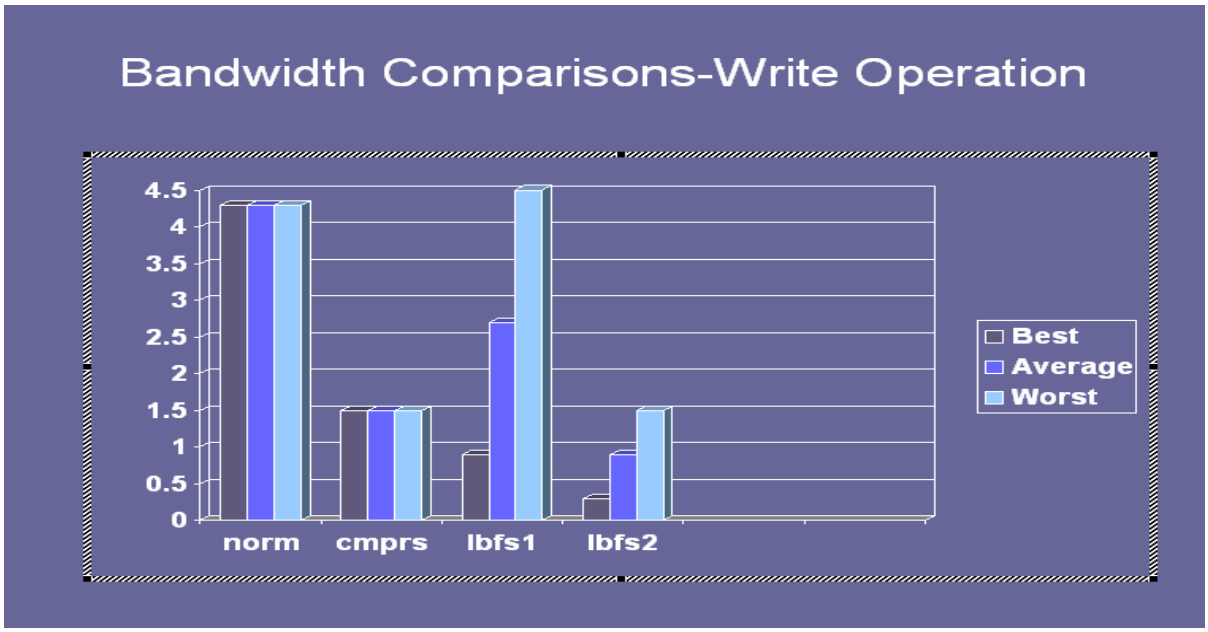

Figure.3 Bandwidth Comparisons-Write Operation

4.3 Disconnected Operation:

Here we have broken the connection purposefully in between the operation & taken the results.

Three types of files are considered

- Video file: 20 MB
- Audio file: 5 MB
- Text file: 2 MB

Following operations are done:

- Norm: file is directly sent from server to client
- Cmprs: file is compressed first & sent from server to client
- Lbfs1- file is divided into chunks & sent using lbfs
- Lbfs2- file is compressed, divided into chunks & then sent using lbfs

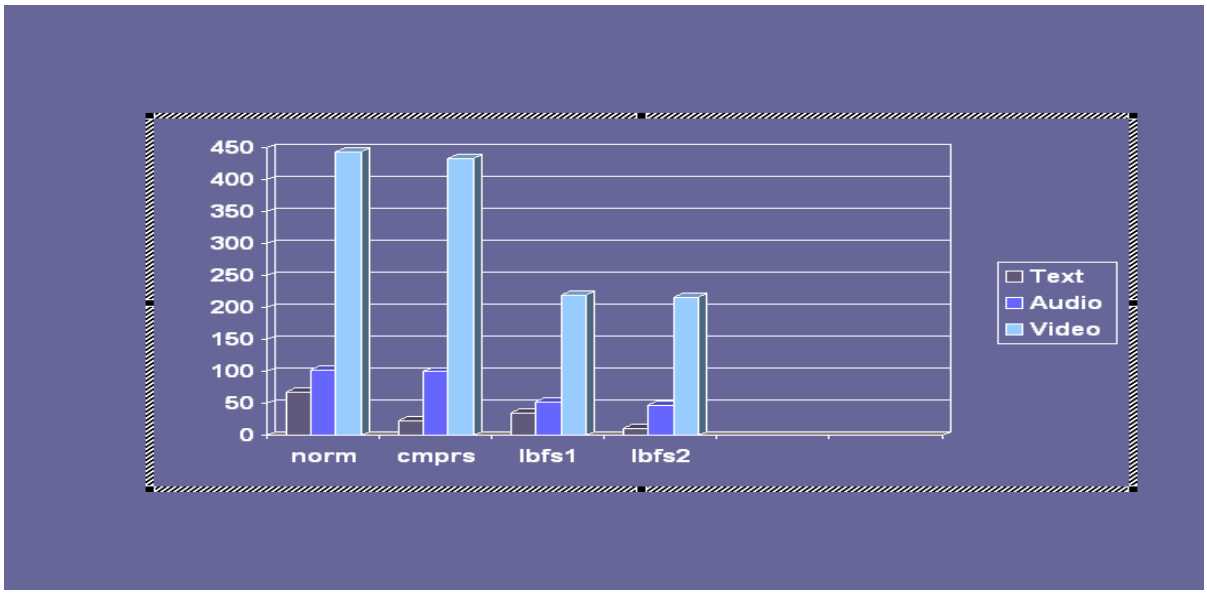

Figure.4 Bandwidth Comparisons-Disconnected Operation

5. CONCLUSION & FUTURE SCOPE

LBFS consumes less bandwidth over network. Also it is quite useful in such situations where network bandwidth is low and data transfer is challenging job, especially in under-privileged areas where available resources are of limited. LBFS leads to optimization of network bandwidth over Internet. This is useful for under-privileged areas where providing Internet facilities is challenging task due to bandwidth problem.

The system that we have developed is very basic but providing maximum LBFS features. The client-server communication can become more interactive. The system can be more interactive. There is scope to incorporate new changes in protocol & naming system. Many

features can be incorporated from other file systems. It is possible to design new LBFS architecture where the LBFS system can be most efficient. It is also possible to create interface layer which supports many types of protocols depends on the applications. The architecture can be scaled to support maximum clients at a time. At the end, it is possible to design a generalized LBFS system which can be used for more generalized applications.

REFERENCES

[1] Athicha Muthitacharoen, Benjie Chen, and David Mazi `eres, "A Low-bandwidth Network File System", MIT Laboratory for Computer Science and NYU Department of Computer Science fathicha, benjieg@lcs.mit.edu, dm@cs.nyu.edu, 2001.

[2] Dan Teodosiu, Nikolaj Bj_ner, Yuri Gurevich, Mark Manasse, Joe Porkka, "Optimizing File Replication over Limited-Bandwidth Networks using Remote Differential Compression", {danteo, nbjorner, gurevich, manasse, jporkka} @ Microsoft.com, Microsoft Corporation, 2006

[3] Val Henson IBM, Richard Henderson Red Hat, "Guidelines for Using Compare-by Hash", Inc. vhenson@us.ibm.com, Inc. rth@redhat.com, 2007

[4] Tim Grembowski1, Roar Lien1, Kris Gaj1, Nghi Nguyen1, Peter Bellows2, Jaroslav Flidr2, Tom Lehman2, Brian Schott2 "Comparative Analysis of the Hardware Implementations of Hash Functions SHA-1 And SHA-512", Electrical and Computer Engineering, George Mason University, University of Southern California, 2002

[5] Andrew S. Tanenbaum, "Computer Networks", Vrije Universiteit, Amsterdam, The Netherlands, 2012